Assignment 1 – LRC Report Template

Arushi Tyagi

CSE 13S – Winter 24

Purpose

The purpose of this program is to replicate the dice game Left, Right, and Center. This multiplayer game consists of players who roll a dice and start off with 3 chips each. If a player rolls a 1, 2, or 3 they get to keep their chip. If they roll a 4, they have to pass one of their chips to the left. If they roll a 5, they put one of their chips in the center. If they get a 6, they give a chip to the player on the right. Whoever is the last person left with any chips, they win.

Questions

Randomness

Describe what makes randomness. Is it possible for anything to be truly random? Why are we using pseudorandom numbers in this assignment?

Randomness is when there are no patterns or predictability when deciding or choosing something. It is difficult for anything to be truly random and highly improbable. We are using pseudorandom numbers in this assignment to create an environment which we can control and also makes it easy to test/debug. With the pseudorandom number, we input a seed which always outputs the same ”random” rolls. That way, we can check to see if the algorithm outputs the right winner and keeps track of each player’s chips correctly corresponding to each seed.

What is an abstraction

When writing code, programmers often use ”abstractions”. Define an abstraction in non computer science terms (Don’t google it!)

Abstraction is focusing on the most relevant parts of code or a program and removing parts that may distract from the main goal. By removing these elements, there is better readability and it is also easier to debug code.

Why?

The last assignment was focussed on debugging. How can abstractions make debugging easier? What other uses are there for abstractions? Hint: Do you have to be the one to write the abstraction?

By removing unnecessary parts of code, we can look at the elements that actually contribute to the output of the program. Then we can look through the program and find the bug in the program easily without any useless aspects. Abstractions are also helpful for testing. When other people want to read or test your code it can make it easier for them to understand what is going on in the program. This helps with the collaboration aspect of coding and creating projects.

Functions

When you write this assignment, you can chose to write functions. While functions might make the program longer, they can also make the program simpler to understand and debug. How can we write the code to use

2 functions along with the main? How can we use 8 functions? Contrast these two implementations along with using no functions. Which will be easier for you? When you write the Program design section, think about your response to this section.

2 functions:
1. Rolling the dice and figuring out where to move the chips
2. Calculating how much each player has and whether there is a winner or if another round needs to be played.

8 functions:
1. Set up game and give everyone 3 chips each
2. Print the name of player rolling the dice
3. Roll the dice (generate a random number)
4. Update what chips everyone has
5. Print how many chips everyone has
6. Check if there is a winner
7. If no winner continue to new player
8. If there is a winner, play the next round

Testing

The last assignment was focused on testing. For this assignment, what sorts of things do you want to test? How can you make your tests comprehensive? Give a few examples of inputs that you will test.

You would need to test whether the user inputs a positive integer or not. Some examples of invalid inputs would be doubles, characters or negative numbers. If they input multiple inputs for the number of players or if there are no inputs. Since the only thing that the user is inputting is how many players are playing and the seed which also needs to be a positive integer, we just need to check this input. For testing the functionality of the program, we would need to make sure that the chips are being moved in the right place based on the dice roll and the math for each player's chip total is being done correctly.

Putting it all together

The questions above included things about randomness, abstractions and testing. How does using a pseudorandom number generator and abstractions make your code easier to test?

With a pseudorandom number generator and abstraction, psuedorandom numbers are reproducible, so by using the same seed the same pseudorandom number will run. Then, we can test it every time the program runs to find where the bug is for that number exactly and debug it easily. With abstractions, the code becomes easier to read as well and we can see directly where the program might be failing.

How to Use the Program

The user will be prompted by the program and will be asked:

```
Number of players (3 to 10)?
```

Then, the user will enter a number and the program will alot everyone with 3 chips. In order to roll the dice, the user will be prompted again and be asked:

```
Random-number seed?
```

The program will print the name of the current player rolling the dice. and the number that has been rolled. Based on the number, the chips will be moved accordingly and the next player will go. This cycle will continue until there is a winner.

Program Design

Audience: Write this section for someone who will maintain your program. In industry you maintain your own programs, and so your audience could be future you! List the main data structures and the main algorithms. You are answering the basic question, "How is this thing organized so that I can have a chance of fixing it?". This section will be longer for a more complicated program and shorter for a less complicated program.

The program would have multiple different functions for better readability and debugging purposes. The first function would take the number of players and seed input and print out the beginning information - the player's name and their roll. They would take their name from the array from names.h. It would then check how many chips each player has. According to this number, it would iterate the roll function 1 time if they have 1 chip, 2 times if they have 2 chips, or 3 chips if they have 3 or more chips. This is indicative of how many rolls they get. The second function takes the roll input and does the according chip movement. The program has an array filled with "dot", "center", "right", and "left". Depending on what the roll is, the chip count for each player would stay the same, increment, or decrement. Then, there would be another function to check if there is a winner or if the program needs to continue to the next person. It would move to the next element/player in the player array and go back to the roll function again.

Pseudocode

Give the reader a top down description of your code! How will you break it down? What features will your code have?

In the first and second functions, I would prompt the inputs for the number of players and the seed value. In my main function, I would create a main while loop. This while loop would iterate through the number of players. Depending on how many chips the current player has, that is the number of turns they get. Based on the number of turns I would iterate through that and call a roll function to get the die roll. This roll function would take the seed number and calculate the corresponding die "number" (DOT, CENTER, LEFT, RIGHT). If the player rolls a dot, then the chips don't move. If it is a center, then the player decrements their chips by one. If it is a right, the previous player gets another chip and the current player loses one. If it is a left, the next player receives a chip and the current player loses one. I would then call the active player function which checks how many active players there are. If there is only one active player then the while loop breaks and the game is over. Then, there would be a printf statement that would print who won the game.

Function Descriptions

For each function in your program, you will need to explain your thought process. This means doing the following

- The inputs of every function (even if it's not a parameter)
- The outputs of every function (even if it's not the return value)
- The purpose of each function, a brief description about a sentence long.
- For more complicated functions, include pseudocode that describes how the function works
- For more complicated functions, also include a description of your decision making process; why you chose to use any data structures or control flows that you did.

Do not simply use your code to describe this. This section should be readable to a person with little to no code knowledge.

- Function: num players. Inputs: number of players, names.h. Output: player name from the names.h array. Purpose: this function takes input from the user to figure out how many players are playing and then returns that number and their respective names.

- Function: seed num. Inputs: seed. Output: seed number. Purpose: creates a separate function for getting the seed from the player for clarity.

- Function: roll. Inputs: the position type. Output: the die roll. Purpose: it takes the seed number and does that number modulus 6. This gives a number from 0 through 5 that would correspond to a face on the die.

- Function: check active players. Inputs: player chip array, number of total players. Outputs: the number of players still playing after each roll. Purpose: computes how many players are still eligible in the game.

- Function: main function. Inputs: number of players, seed number, roll, active player count. Outputs: the number of chips each person has after every roll. Purpose: in the main function, all the other functions are called to calculate how many chips every player has. The function checks which player is rolling and how many turns they get. Based on each roll, the number of chips each player has changes. It uses the active player function to determine whether there is a winner or not.

References